1. **What do you mean by Entity- Relationship Diagram? Explain**

An entity-relationship diagram is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within that system. An E-R diagram is a [data modeling](http://searchdatamanagement.techtarget.com/definition/data-modeling) technique that can help define business processes and can be used as the foundation for a [relational database](http://searchsqlserver.techtarget.com/definition/relational-database).

While useful for organizing [data](http://searchdatamanagement.techtarget.com/definition/data) that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semi-structured or [unstructured data](http://searchbusinessanalytics.techtarget.com/definition/unstructured-data), and an E-R Diagram is unlikely to be helpful on its own in integrating data into a pre-existing information system.

Three main components of an ERD are the [entities](http://whatis.techtarget.com/definition/entity), which are objects or concepts that can have data stored about them, the relationship between those entities, and the [cardinality](http://whatis.techtarget.com/definition/cardinality), which defines that relationship in terms of numbers.

For example, an ER diagram representing the information system for a company's sales department might start with graphical representations of entities such as the sales representative, the customer, the customer's address, the customer's order, the product and the warehouse. (See diagram) Then lines or other symbols can be used to represent the relationship between entities, and text can be used to label the relationships. *[1]*

1. **Define entity and give an example.**

In general, an entity is an existing or real thing.  In programming, engineering, and probably many other contexts, the word is used to identify units, whether concrete things or abstract ideas, that have no ready name or label. In blackboard discussions, one can draw something as yet unnamed and refer to that drawing as the representation of an entity. If the entity being discussed later gets ascribed qualities and a name, reference to it as an "entity" may no longer be useful.

***Example***

* In relation to a [database](http://searchsqlserver.techtarget.com/definition/database) , an entity is a single person, place, or thing about which data can be stored.
* In [data modeling](http://searchdatamanagement.techtarget.com/definition/data-modeling) (a first step in the creation of a database), an entity is some unit of data that can be classified and have stated relationships to other entities.
* In the Open Systems Interconnection ( [OSI](http://searchnetworking.techtarget.com/definition/OSI) ) model of network communication, an entity is an active element within a subsystem that communicates with other entities using a defined protocol.*[2]*

1. **Explain the different between an entity class and an entity instance.**

|  |  |
| --- | --- |
| **Entity Class** | **Entity Instance** |
| A class may have zero or more instances | An instance may only have one class. |
| A class may have one or more super-classes (parents). | A instance can only have one parent. |
| Entities that have the same attributes are grouped in what are best called entity classes (it doesn't make sense to refer to a collection of entities as an entity) | If an entity is an individual "person, place, event, or thing about which data is collected", then an entity is an instance |

**4. Define attribute and its types.**

Attributes are, simply put, the characteristics of entities. Some entities can have many attributes while others may only have a couple. As well, there are five categories that attributes are classified in. This simple table will be used to explain how each attribute can be a different type of attribute:

Required or Optional Attributes

A required attribute is an attribute that must have a value in it, while an optional attribute may not have a value in it and can be left blank. The reasoning for making an attribute required is to put emphasis on what is important in that entity and what makes it stand out from other entities.

Example: Consider the entity Student abovestud\_LastName and studFirstName would be required attributes as it uniquely defines that table and we assume all students have a first and last name. Optional attributes in the table Student could be stu\_MiddleName, stu\_Email, and stu\_Phone since some students may not have a middle name, a phone number, or an email address.

Keys and non-keys Attributes

In every entity an attribute or grouped attributes uniquely identify that entity. These attributes are the key attributes and range from Primary key (single attribute identifier) to a Composite Key (Multi attribute Identifier). The rest of the attributes after the identifier are considered the non-key attributes or descriptors, which just describe the entity.

Example: Above in the table Student there is only one unique identifier, stu\_LastName, which is the primary key of the table. The rest of the attributes are descriptors.

Single and Composite Attributes

Attributes can be classified as having many parts to them or just a single unbreakable attribute. The composite attribute is an attribute that can be subdivided into other single attributes with meanings of their own. A single attribute is just an attribute that cannot be subdivided into parts.

Example: Imagine from the entity Student that instead of having the three attributes: stu\_LastName, stu\_MiddleName, stu\_FirstName it had one attribute called stu\_Name. The attribute stu\_Name would be considered a composite attribute since it can be subdivided into the other three attributes: stu\_LastName, stu\_MiddleName, stu\_FirstName. The rest of attributes would be consider single attributes since they can't be subdivided into parts

Single-valued and multi-valued Attributes

Attributes can be classified as single or multi-value. The single-value attribute can only have one value, while the multi-valued attributes usually can store multiple data in them.

Example: In the entity Student, stu\_Address could be considered a multi-value attribute since a student could have multiple addresses where he lives at. An example of a single-value attribute would be stu\_LastName since a student usually has one last name that uniquely identifies him/her.

Derived Attributes

The last category that attributes can be defined is called a derived attribute, where one attribute is calculated from another attribute. The derived attribute may not be stored in the database but rather calculated using algorithm.

Example: In the entity Student, stu\_Age would be considered a derived attribute since it could be calculated using the student's date of birth with the current date to find their age.[4]

1. **What is derived attributes?**

Attributes can be defined is called a derived attribute, where one attribute is calculated from another attribute. The derived attribute may not be stored in the database but rather calculated using algorithm.

Example: In the entity Student, stu\_Age would be considered a derived attribute since it could be calculated using the student's date of birth with the current date to find their age.[4]

1. **Define relationship and give an example.**

A relationship, in the context of databases, is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table. Relationships allow relational databases to split and store data in different tables, while linking disparate data items.

For example, in a bank database a CUSTOMER\_MASTER table stores customer data with a primary key column named CUSTOMER\_ID; it also stores customer data in an ACCOUNTS\_MASTER table, which holds information about various bank accounts and associated customers. To link these two tables and determine customer and bank account information, a corresponding CUSTOMER\_ID column must be inserted in the ACCOUNTS\_MASTER table, referencing existing customer IDs from the CUSTOMER\_MASTER table. In this case, the ACCOUNTS\_MASTER table’s CUSTOMER\_ID column is a foreign key that references a column with the same name in the CUSTOMER\_MASTER table. This is an example of a relationship between the two tables.[5]

1. **Explain the difference between a relationship class and a relationship instance**.

**8. Define degree of relationship.**

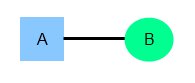
Degree of relationship refers to the number of participating entities in a relationship. If there are two entities involved in relationship then it is referred to as binary relationship. If there are three entities involved then it is called as ternary relationship and so on.

On the other hand, it is the cardinality of relationship that defines the number of instances of one entity as it relates to the number of instances of the other entity. Based on the different combinations between two entities we can have either one-to-one, one-to-many or many-to-many relationship. [6]

**9. List and give an example of the three types of binary relationships. Draw an E-R diagram for each.**

## *1. One-to-One*

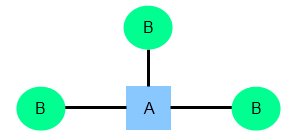
One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).



For example, if each customer in a database is associated with one mailing address.

## *2. One-to-Many*

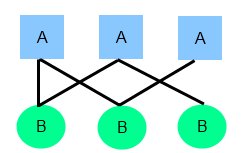
One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.



For example, a single customer might place an order for multiple products. The customer is associated with multiple entities, but all those entities have a single connection back to the same customer.

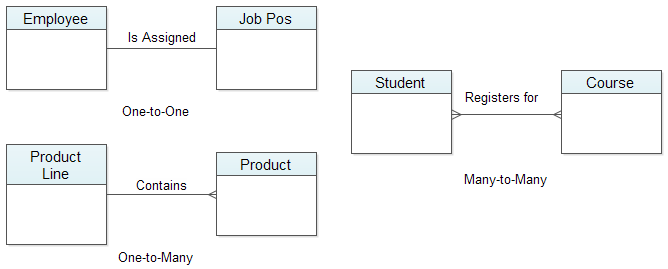
## *3. Many-to-Many*

One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.



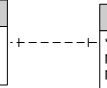
For example,at a company where all call center agents work with multiple customers, each agent is associated with multiple customers, and multiple customers might also be associated with multiple agents.[7]

More E-R Diagram



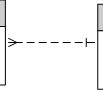
**10. Define the terms maximum cardinality and minimum cardinality.**

**Maximum cardinality** indicates how many instances are participating in a relationship. The possibilities include one-to-one (1:1), one-to-many (1:M), or many-to-many (M:N). A 1:1 relationship can be thought of as the relationship between a football stadium and the home team. There can be only one team per stadium. You would denote this by having a perpendicular dash next to each entity in the relationship.



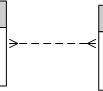
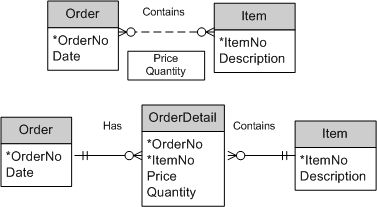
1:1 Relationship

A 1:M or M:1 relationship, involves one instance (the parent) of an entity in a relationship that connects to many instances (the children) in the other entity. This is the most common relationship seen in the relational database. Logically it should make sense, for example a single Course has many Classes/Sections, one Employee takes/instructs many Classes, or one Building has many Rooms. You illustrate the many by using a crow's foot.



M:1 Relationship (read left to right)

A M:N relationship can be used in a conceptual model to illustrate a situation where many instances of one entity in a relationship with many instances of the other. In the second solution in the previous page, you see that Employees can take many Courses but a Course also contains many Employees. This relationship also had an attribute Attendance, M:N relationships may have additional information that is stored as part of the instance of the relationship rather than with either entity. If you imagine an order and an item, the order contains many items and an item can be sold on many different orders. For and instance of Order-Item, there can be a price or quantity that corresponds to a specific instance of the relationship so quantity and price belong to the relationship rather than either item (see the equivalent pictures below). To avoid that in the first solution, an additional table was added. I will accept either approach a M:N with attributes or an additional entity (called an associative entity) unless otherwise noted.

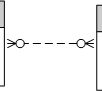
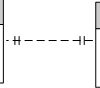
M:N Relationship Equivalent conceptual designs, the composite identifier in OrderDetail uniquely identifies an items

quatity and price for a specific order

 A many can also be labeled as finite. Imagine a classroom with 36 seats. See below the example below.

finiteM.jpg

The **minimum cardinality** indicates the smallest number of participants in a relationship, which can be 0 or 1 (optional or mandatory). When evaluating minimum cardinality, you should think about what is actually taking place. Rarely is there a situation that is mandatory-to-mandatory (difficult to implement because you are stating the instances must both exist simultaneously) or optional-to-optional (an "open design," usually shown with a M:N), rather it is some form of optional-to-mandatory or mandatory-to-optional. For example, you can read that a building must conceptually be mandatory for a room to exist, but the building can exist without rooms. Ultimately, you are defining the order of adding data to your database. The building instance must be in the database before any room instances. The inner marks indicate minimum cardinality below.

Strong M:N, optional-to-optional Weak1:M, mandatory-to-optional Strong 1:1, mandatory-to-manadatory . [8]

**References**

**[1].** [**http://searchcrm.techtarget.com/definition/entity-relationship-diagram**](http://searchcrm.techtarget.com/definition/entity-relationship-diagram)

**[2]** [**http://whatis.techtarget.com/definition/entity**](http://whatis.techtarget.com/definition/entity)

**[3]** [**http://www.openresource.com/on\_ontiki/CIER.php /**](http://www.openresource.com/on_ontiki/CIER.php%20%20%20/)[**http://www.answers.com/Q/Difference\_between\_an\_entity\_class\_and\_an\_entity\_instance**](http://www.answers.com/Q/Difference_between_an_entity_class_and_an_entity_instance)

**[4]** [**http://databasemanagement.wikia.com/wiki/E/R\_Model:\_Type\_of\_Attributes**](http://databasemanagement.wikia.com/wiki/E/R_Model:_Type_of_Attributes)

**[5]** [**https://www.techopedia.com/definition/24438/relationship-databases**](https://www.techopedia.com/definition/24438/relationship-databases)

**[6]** [**http://www.geekvillage.com/forums/showthread.php?t=34252**](http://www.geekvillage.com/forums/showthread.php?t=34252)

**[7]** [**http://www.webopedia.com/TERM/E/entity\_relationship\_diagram.html**](http://www.webopedia.com/TERM/E/entity_relationship_diagram.html)

**[8]** [**http://www.itk.ilstu.edu/faculty/bhosack/378M2\_ERD/378Mod2ERD5.html**](http://www.itk.ilstu.edu/faculty/bhosack/378M2_ERD/378Mod2ERD5.html)